Final Project

Navigation Competencies

Read this entire project before starting to work on the final project. Please make sure to ask questions if anything is unclear.

Reading and Review:

Ch. 4 of the text and Weeks 1 through 7 lectures

Purpose:

The purpose of the final project is for the student team to demonstrate the integration of several concepts learned this quarter. The primary goal is to develop several navigation competencies on the robot. To achieve the navigation tasks, the student team will use wireless communication and a graphical user interface to communicate data between the robot and team laptop.

Objectives:

At the end of this project, the student should be able to:

- Implement light tracking on a reactive controller and as an added layer on a subsumption architecture with random wander and avoid obstacles
- Use wireless communication to send and receive data to the robot from the computer
- User wireless communication to transmit data between robot, computer and a GUI (Human-Robot Interface)
- Use sensor feedback to estimate the robot's pose in the world
- Implement topological path following on a mobile robot to move the robot from a start point to a goal location
- Implement metric path planning on a mobile robot by using wave front propagation or grassfire expansion to move the robot from a start to a goal location
- Use sensor data and an exploration algorithm to localize a robot in a world given an a priori map
- Use sensor data and a coverage algorithm to create an occupancy grid or topological map of the robot's world given a robot's known pose
- Implement simultaneous localization and mapping (SLAM) to locate a beacon in an unknown world and then escape from the world when a wall suddenly disappears. Use the camera to identify landmarks in the world and put their location on the created map or use to localize the robot.

Equipment:

- Base Robot
- Encoders
- IMU (Inertial Measurement Unit) - MPU6050 Module 3 Axis Analog Gyro + 3 Axis Accelerometer Module
- Wireless Controller (WiiMote, PlayStation Controller, Keyboard)
- Flashlight
- 2 photo resistors
- 2 resistors
- IR Remote
- Masking Tape
- Ruler
- Various Wires
- 3 LEDs

References:

- MATLAB Support Package for Arduino
  http://www.mathworks.com/help/supportpkg/arduino/examples.html
  Simulink Support Package for Arduino
  http://www.mathworks.com/help/supportpkg/arduinoio/examples.html
- SimuLink and Arduino Getting Started Guide:
  http://makerzone.mathworks.com/resources/install-support-for-arduino/?s_eid=PRP_5807
- Processing Resources
  https://processing.org/reference/
- HC05 Bluetooth Module tutorial
  https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/
- MPU6050 IMU tutorial
  https://maker.pro/arduino/tutorial/how-to-interface-arduino-and-the-mpu-6050-sensor

****

THEORY

****

Navigation in mobile robotics deals with several questions that you will answer in completion of the final project. These questions are:

- Where am I going? (Path Planning)
- How do I get there? (Path Following)
- Where am I? (Localization)
- Where have I been? (Mapping)

Light Tracking

The purpose of this lab is to use two photo resistors connected to your robot to implement light sensing behaviors. The light sensor will be used to implement a reactive controller similar to Braitenberg's vehicles 2 and 3. The light sensing will then be added as a layer to the robot's subsumption architecture and integrated into the random wander and obstacle avoidance state machine.

The second purpose is to use a type of locomotion called ***homing*** or ***docking*** with hybrid control to move the robot toward a light source. There will be a static light source placed in the environment which the robot can easily sense. The goal will be for the robot to move toward the beacon and stop just before hitting it. There will be no fixed path to the beacon, the robot should follow walls until the beacon is sensed, it should then leave the wall, keep track of its state, and use the move to goal behavior to dock on the source while avoiding any obstacles along the way. Lastly, the robot should then turn 180 degrees and return to the wall to continue following as near as possible to the spot where it left. This will only be possible if the robot has kept track of its state.

Photoresistor for light sensing

A photoresistor is a semiconductor device whose resistance is a function of light intensity. The schematic symbol for the photoresistor is shown in Figure 1. Because the resistance of the photoresistor varies with light intensity, the current that flows through it also varies with light intensity. However, we want to monitor voltage, not a current, since the ADC (Analog-to-Digital Converter) on the micro-controller takes voltage measurements. We will be able to monitor voltage from the photoresistor by creating a simple voltage divider circuit, as shown below.

The photoresistor used in this lab is designed to have maximum resistance in the absence of light. As light intensity increases, its resistance decreases. As a result, as light intensity increases, the voltage, V_o, in the voltage divider circuit will also increase. You will monitor this voltage V_o as a measure of the light intensity seen by the photoresistor. On the robot, V_{ref} is the +5V supply and V_o is connected to an analog pin on the microcontroller.

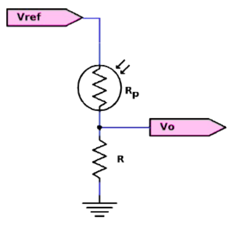


Figure 1: Photoresistor Wiring

In this configuration, the output voltage is given by,

$$V_o = \left(\frac{R}{R + R_p}\right) V_{ref}$$

Since the lighting environment and photoresistors will vary, your first task will be to measure the voltage output of your photoresistor for various dark and light settings.

Topological Path Following

Topological path following is based upon landmarks in the world. A ***distinctive place*** is a landmark where the robot can make a navigation decision. Typically, the robot will use one behavior to move in the world and then change when it gets close or in the neighborhood of the distinctive place. For the purpose of this project, the distinctive places in the world will be hallways, corners, t-junctions, and dead ends. The student team will design behaviors and perceptual schema for identifying gateways in an artificial environment. The robot will be given a list of navigation commands based upon the world's topology and use a parsing routine and a sequencer to move the robot from a start point to a goal point. For example, if the robot is given "SLRT" (S = **S**tart, L = go **L**eft, R = go **R**ight, T = **T**erminate) then it would start, follow hallway, turn left at the next gateway, turn right at the next gateway, then stop at the last gateway. The robot should continue to move forward until a gateway is encountered or until the stop command is encountered. Figure 2 is an example of topological navigation using the command "SRLT".

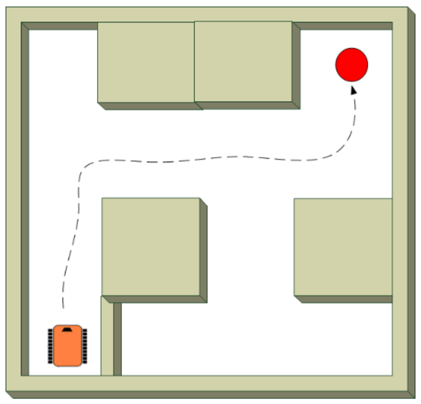
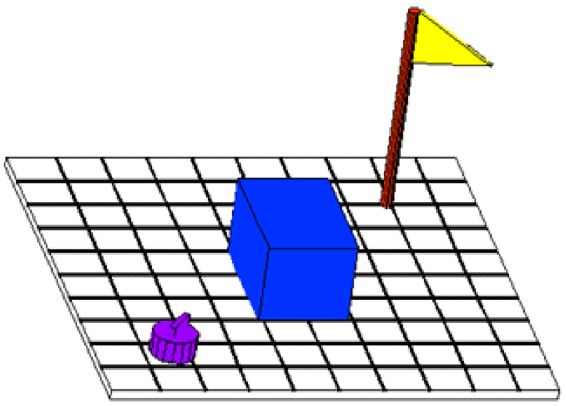


Figure 2: Topological Navigation ("SRLT")

Metric Map Path Planning and Execution

In this project, you will use a wavefront algorithm (or grassfire expansion) on an a priori map to create a path from the robot's start position to goal location. Use the wall following, obstacle avoidance and move to goal behaviors to move through the list of goal points until the robot arrives at the final destination. If you assume the algorithm uses an eight-neighborhood, the robot can move diagonally, otherwise a four-neighborhood would be used which may have less odometry error. The configuration space will be an occupancy grid divided into 18" x 18" squares, where free space is represented by 0's and occupied space by 99's. You should devise a scheme to represent the robot's start position and goal position. Your code should be flexible such that these values can be specified at run time. Figure 3 is an example of the world representation.



0	0	0	0	0	0	0	0
0	0	0	0	0	0	G	0
0	0	0	0	0	0	0	0
0	0	0	99	99	99	0	0
0	0	0	99	99	99	0	0
0	0	0	99	99	99	0	0
0	0	0	0	0	0	0	0
0	S	0	0	0	0	0	0

a. Real world b. Configuration Space (8 x 8 matrix)

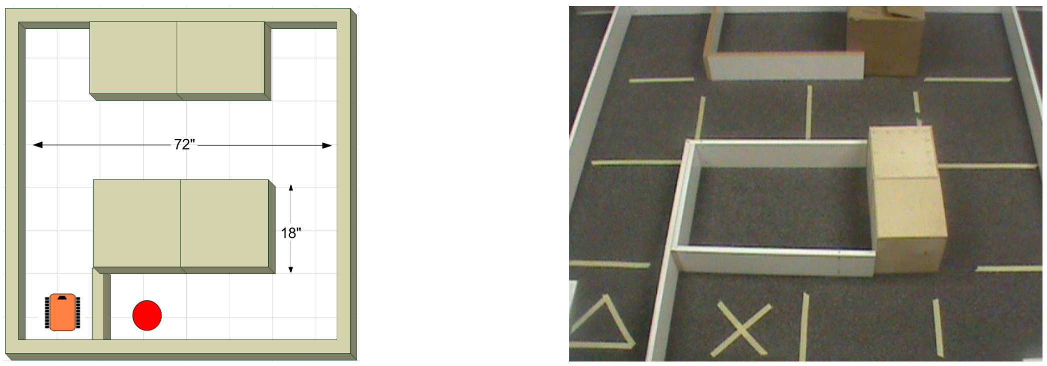
Figure 3: World Representation

The wavefront is created by starting at the destination and creating eight connected neighbors back to the start point. The numbers represent the number of steps to the goal point. The robot would then follow the numbers in the reverse order to arrive at the goal point (see Figure 4). The goal is for the robot to always move such that the steps to the goal position are reduced. This path plan has 9 steps from start to finish. Note that you may need to grow the obstacles by the robot’s width or radius to avoid clipping them.

The test arena for the project demonstration will be 6 ft x 6 ft with 18” x 18” obstacles. This artificial world will be a 4 x 4 grid where the robot’s start point is denoted by a ‘Δ’ and the goal point is marked by an “X”. Figure 5 shows a sample test arena.

7	6	5	4	3	2	1	1	1
7	6	5	4	3	2	1	0	1
7	6	5	4	3	2	1	1	1
7	6	5	99	99	99	2	2	2
7	6	6	99	99	99	3	3	3
7	7	7	99	99	99	4	4	4
8	8	8	7	6	5	5	5	5
9	9	8	7	6	6	6	6	6
10	9	8	7	7	7	7	7	7

Figure 4: World Representation Wavefront



a. Artificial world b. Real world

Figure 5: Sample Test Arena

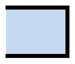
Instead of representing the world map as an occupancy grid, it can be based upon the topology of the space. The salient features of the space are walls, hallways, corners and junctions. Each square will be represented by an integer between 0 and 15, dependent upon where walls are present around the square. The north (0001), east (0010), south (0100) and west (1000) walls represent one bit of that integer (see Table A). Using the coding in Table 1, the maze shown in Figure 5 is represented by an 8 x 8 matrix of integers shown in Figure 6. Using the coding in Table 1, the maze shown in Figure 7 is represented by an 11 x 10 matrix of integers.

To use the topological map to plan a path from a robot start location to a goal point it is possible to use the wavefront algorithm again. However, instead of the robot moving to cells on the occupancy grid, the robot will use behaviors and rules such as move forward, turn left, follow wall, follow hallway, avoid obstacles, etc. The key difference between this path following and the first one described is there are walls in the world as opposed to occupied or empty cells. This navigation involves taking the list of actions and executing them.

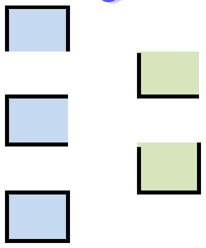
9	3	15	15	15	15	9	3
8	2	15	15	15	15	8	2
8	0	1	1	1	1	0	2
8	0	4	4	4	4	0	2
8	2	15	15	15	15	8	2
8	2	15	15	15	15	8	2
8	2	9	1	1	1	0	2
12	6	12	4	4	4	4	6

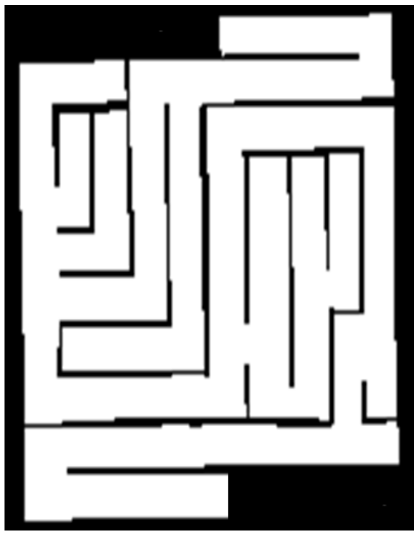
Figure 6: Sample Test Arena Topological Map

Table A: Topological map coding

Integer	Binary	Hexadecimal	Direction	Wall Location
0	0000	0		
1	0001	1	North	
2	0010	2	East	
3	0011	3		
4	0100	4	South	
5	0101	5		
6	0110	6		
7	0111	7		
8	1000	8	West	
9	1001	9		
10	1010	a		

11	1011	b
12	1100	c
13	1101	d
14	1110	e
15	1111	f





15	15	15	15	15	13	5	5	5	3
9	5	7	9	1	5	5	5	5	6
10	11	11	10	10	9	5	5	5	3
10	10	10	10	10	10	11	11	11	10
8	6	10	10	10	10	10	10	10	10
8	5	6	10	10	10	10	10	10	10
8	5	5	6	10	10	10	10	6	10
10	13	5	5	6	8	2	10	9	2
12	5	5	5	5	6	12	6	10	14
9	5	5	5	5	5	5	5	4	7
12	5	5	5	5	15	15	15	15	15

Figure 7: Maze Topological Map

Localization

Localization is the process a robot uses to determine its location in a world given an a priori map and sensor readings. A localization algorithm should process the map to identify key features such as the gateways or distinctive places [corners (C), hallways (H), dead-ends (D), t-junctions (T)]. Although it is a not a gateway, a series of 10's or 5's or neighboring 1's and 4's or 8's and 2's indicates a hallway (H) in the world. The numbers (1, 2, 3) indicate a gateway where the robot can make a navigation decision. The distinctive features or gateways are the nodes in the world and the hallways can be used with a local control strategy such as follow center or follow wall to move between nodes. An example of coding a map for topological path following is shown in Figure 8.

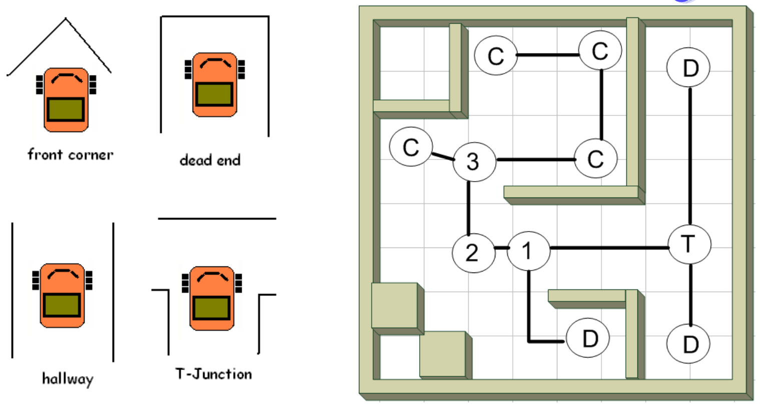


Figure 8: Feature Extraction (Map Coding)

Once the robot is placed in the world, the robot should then use some motion algorithm such as random wander, follow center, or wall following to explore the world and identify gateways. It should keep track of the gateways passed and the order in which they were passed. Although there will be some odometry error, it would also aid the localization algorithm to keep track of odometry such as distance traveled and turns. Within three to four iterations of this process, the robot should be able to use a probabilistic method such as the Partially Observable Markov Decision Process (PMDP) to localize itself. Note that if the robot also uses directional information such as it starts out facing north in in the world, it can reduce the number of steps required to localize. Figure 9 provides an example of this localization process with the proposed robot locations after each step marked on the map.

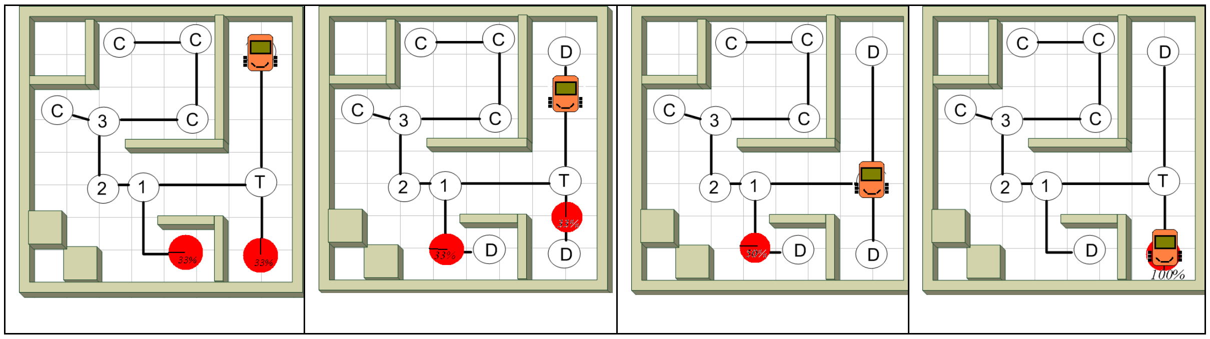


Figure 9: PMDP Localization

Mapping

Mapping is the process that a robot uses with sensor feedback to create a map of an unknown environment. It is possible to use Histogrammic in motion mapping (HIMM) to identify objects in the environment. Although your textbook states that this algorithm was created for a sonar sensor, it is reasonable to use it for the primary axis of the IR sensor. For this method when the cell value exceeds some threshold it is coded as occupied or used to create a topological map. Alternately, you can create an occupancy grid with 0's and 99's based upon the free and occupied space. Figure 10 provides an example of using a Generalized Voronoi Graph (GVG) and HIMM to create a world map.

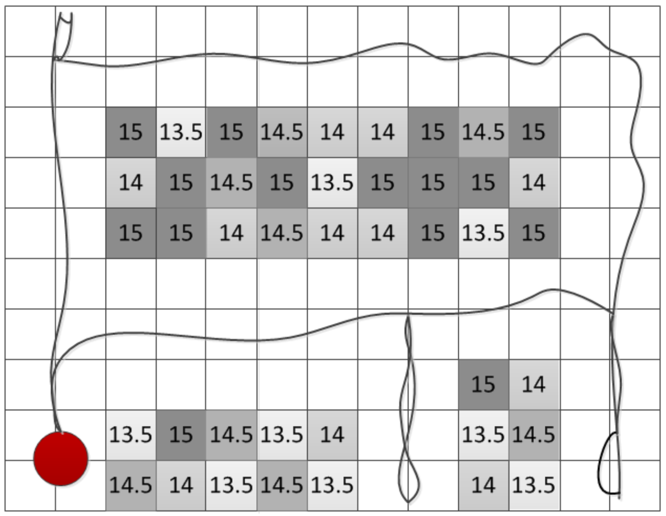


Figure 10: GVG with HIMM

****

PROCEDURE

****

Part A1 – Light Tracking

1. Face the sonar sensors over the front right and left corner of the robot and mount 2 photoresistors to analog pins on the front of the Arduino Robot. The sensors should face forward with one on the right in line with the right motor and one on the left in line with the left motor. You may need to use electrical, painters, or masking tape to anchor them to the desired location. If you have problems mounting your sensors, please go see the ECE technicians for help.

2. Read the analog value of the photoresistors to calibrate for dark and light values. You will use these values to account for ambient lighting and help you implement the reactive

controllers. Complete Tables 1 and 2 for both sensors to give you some idea of how to use them for the next part. ***You should include these tables in your lab worksheet submission.***

Table 1: Environment Data

Conditions	Left Photoresistor (V)	Right Photoresistor (V)
Ambient light on the table		
Ambient light under the table		
Sensor covered		
In front of a flashlight or cell phone light		

Table 2: Distance and Angle of Incidence Data

		Left Photoresistor (V)			Right Photoresistor (V)		
Environment	Distance (in)	Angle of Incidence -45°	Angle of Incidence 0°	Angle of Incidence 45°	Angle of Incidence -45°	Angle of Incidence 0°	Angle of Incidence 45°
On the table	6						
On the table	12						
On the table	18						
On the table	24						
On the table	30						
Under the table	6						
Under the table	12						
Under the table	18						
Under the table	24						
Under the table	30						

Part II - Reactive Control

1. The first program you will write is a reactive controller inspired by Braitenberg's vehicle experiments. In this step, you will create a vehicle that is wired with excitatory connections where each sensor is connected to the motor on the same side. The program controls the left and right wheels based upon the light intensity seen by the left and right photoresistors (see Figure 11a). Turn on the RED, YELLOW, and GREEN LEDs when the behavior is active. ***The default motion for the robot is stopped until the light is sensed.***

2. How does the robot behave when (a) the light source is directly in front of the robot, (b) the light source is to one side of the robot? Is there anything about the robot's behavior that surprises you? ***Answer this question in the lab worksheet.***

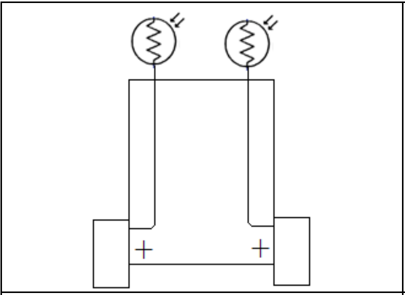
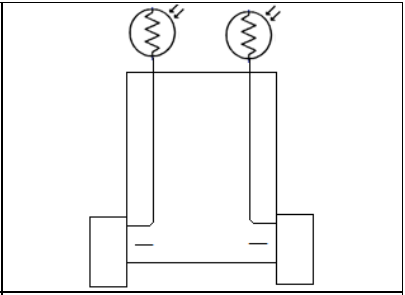
3. Next, repeat parts 1 and 2 except that each sensor is connected in an inhibitory manner. This means the motor slows down as it gets closer to the light (see Figure 11b). Turn on the RED and YELLOW LEDs when the behavior is active. ***The default motion for the robot is driving forward slowly until the light is sensed.***

4. Next, repeat parts 1 and 2 except cross the connections between the motors and the sensors so that the left light sensor controls the right motor's speed and vice versa in an inhibitory manner (see Figure 11c). Turn on RED and GREEN LEDs when the behavior is active. ***The default motion for the robot is driving forward slowly until the light is sensed.***

5. Finally, repeat parts 1 and 2 with the connections still crossed between the motors and the sensors so that the left light sensor controls the right motor's speed and vice versa in and excitatory manner (see Figure 11d). Turn on the GREEN and YELLOW LEDs when the behavior is active. ***The default motion for the robot is stopped until the light is sensed.***

6. Braitenberg called these four light sensing behaviors, fear, aggression, love, and explorer. These are the emergent behaviors that you did not explicitly program. Can you identify which of the four behaviors (fear, aggression, love, explorer) is exhibited for each of the prior motor/sensor connections? ***Answer this question in the lab worksheet.***

7. How did you decide on the position of the photoresistors? Were there certain lighting conditions that were more difficult or easier for the robot to sense? ***Answer this question in the lab worksheet.***

	
(a)	(b)

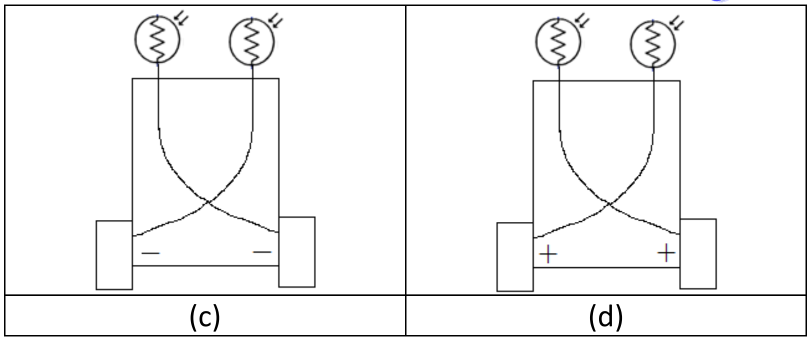


Figure 11: Valentino Braitenberg Vehicles

Hints: To get variable speed on the robot stepper motors, you need to use **runSpeed()** or **runSpeedToPosition()**. You need to set the speed right just before you call these functions. Set the right and left motor speed proportional to the light. You will need to call **runSpeed()** and **runSpeedToPosition()** inside of the main loop as frequently as possible to see the variation. Do not use the **RunToStop()** function. You must use setMaxSpeed if setSpeed does not work to change the motor speed.

Part III - Obstacle Avoidance

1. After testing each of the sensorimotor connections individually and confirming that they work correctly, you should add this light following behavior as layer on a subsumption architecture.
2. Layer 0 of the architecture should be avoid obstacle with collide and run away, layer 1 should be random wander with an input to avoid. Layer 2 should be light following which subsumes the output of random wander.
3. If the robot detects a light it should move with respect to the sensorimotor connections toward the light source while also avoiding obstacles. If the robot does not detect a light source or an obstacle, then it should random wander. Figure 12 is an example of a subsumption architecture, but it does not include random wander. Turn on RED LED to indicate an obstacle. Turn on a GREEN LED to indicate random wander.

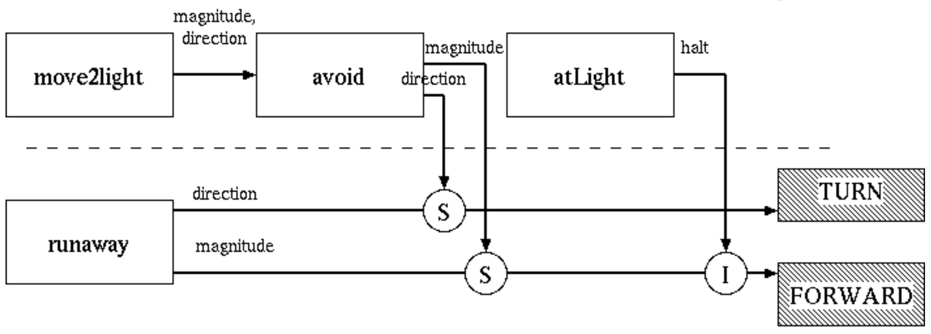


Figure 12: Sample Photophilic Architecture (no random wander)

Part IV – Homing or Docking

1. Next, you will design a completely different architecture for hybrid control. The hybrid control architecture that you will implement to home the robot includes a reactive layer (obstacle avoidance, wall following, move to goal (light following), path update), middle layer (arbitrator), and deliberative layer (current state, path plan, follow back to wall). This architecture is shown in Figure 13. Your code should be written in a modular fashion with functions such that it is evident where the planning, sensing and acting take place. Turn on a GREEN LED to indicate wall following.

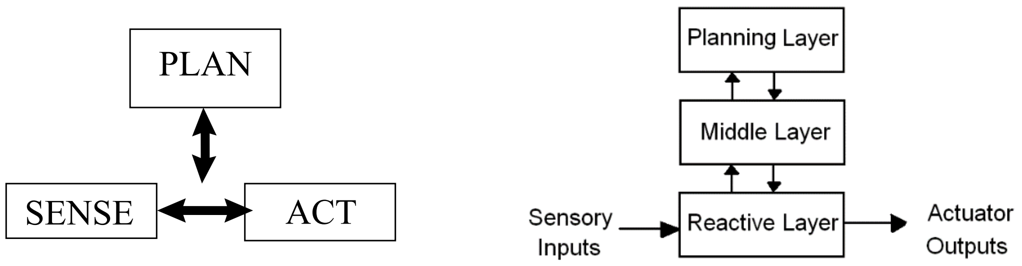


Figure 13: Homing Hybrid Control Architecture

2. The partial world map (representation) includes direction to the beacon and back to the wall with respect to the robot's current pose. This representation will be input into the deliberative layer for path planning. Updates to the path will be based upon feedback from the distance, heading and photoresistors. The middle layer will be used to make decisions about whether path updates are handled in the deliberative or reactive layer. The reactive layer will handle obstacle avoidance, wall following and move to goal behaviors. The robot should turn around and follow the path to drive back to the wall.

3. Based upon the above model, write code to home the robot to the light source (see Figure 14). The robot should come within one foot of the beacon without touching it. Turn on a YELLOW LED to indicate homing (moving to the goal, light following) and path planning (updates). Turn on a RED LED to indicate path following.
4. Test your final control algorithm for several different robot start points or beacon locations and summarize the results in your lab worksheet.

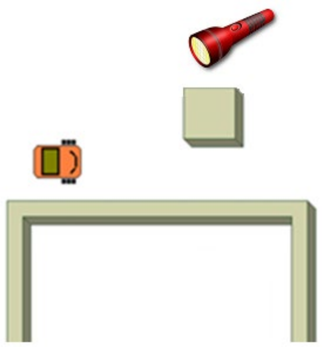


Figure 14: Robot homing

Part V – Docking the Robot and Return to the Wall

Improve the homing routine implemented in the previous part by docking the robot (back to the light) (see Figure 15). The robot should then follow the original path back to the wall to continue wall following. It is your choice which the direction the robot goes once it returns to the wall. Turn on the RED LED to indicate when the robot is path following back to the wall.

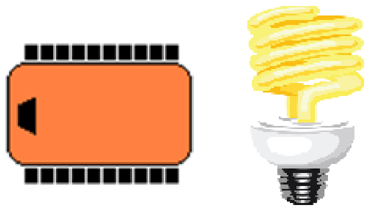


Figure 15: Robot Docking

Part A2 – Graphical User Interface (with/without wireless communication)

1. Review the following links to learn how to use Processing, MATLAB, JAVA, or Python to create a GUI (Human-Robot Interface) for your robot. It will use the Bluetooth module and

serial communication to send and receive data with the robot and laptop. The team can use any language they are comfortable with for this part.

- http://www.sojamo.de/libraries/controlP5/#examples
- https://www.youtube.com/watch?v=NTXkmpfTklQ
- https://www.youtube.com/watch?v=5WjEQSMiqMQ&t=506s
- https://www.youtube.com/watch?v=Rgq8oy3to-E&t=10s
- https://www.mathworks.com/help/supportpkg/arduinoio/index.html?s_tid=CRUX_lftnav
- https://www.mathworks.com/help/matlab/creating_guis/about-the-simple-guide-gui-example.html
- https://www.mathworks.com/help/supportpkg/arduinoio/getting-started-with-matlab-support-package-for-arduino-hardware.html?s_tid=CRUX_lftnav
- Building Apps with MATLAB & APP DEsigner

https://www.youtube.com/watch?v=nQb0qBiDurU&ab_channel=MATLAB

- Create and Run a Simple App Using App Designer

https://www.mathworks.com/help/matlab/creating_guis/create-a-simple-app-or-gui-using-app-designer.html

2. Design a user interface that at a minimum includes all of the following features

- User can remotely drive the robot through the interface
- User can enter topological paths to send to robot
- User can send start and goal positions to the robot
- GUI displays sensor data (sonar, IR, IMU, encoders)
- GUI displays the robot world with planned path
- GUI shows robot status such as task progress, current localization, map created, beacon found, exit found, escape status

Question to answer in the final report:

- Describe how you made design choices for the robot graphical user interface.

- Describe how you implemented wireless communication on the robot
- Create a mockup of the GUI you will design to send data and display sensor data for path planning, path following, localization and mapping information, and SLAM if you do that part. Include it in the appendix of the final report and discuss it in the results section.

PART B – PATH FOLLOWING & PATH PLANNING

Part B.1 - Topological Path Following

1. The student team should place the robot in several corners and intersections of hallways in the artificial environment and determine the perceptual schema to identify these gateways and distinctive places.
2. It would be advisable to use a combination of the IR and sonar for sensor redundancy to identify these locations in the world. There will be some sensor error, this is a standard problem in mobile robot navigation and the program should be designed in such a way to minimize the effect of the error. For example, average 5 or 10 readings and filter out spurious readings. Figure 16 provides descriptions of the possible world landmarks.
3. Include a table similar to Table 3 that includes the test data and perceptual schema for each of the landmarks. You should have a minimum of 4 sensors around the perimeter of the robot in order to get sufficient data for mapping and localization. You may also have a combination of IR and sonar.
4. Program the robot to identify the gateways and make navigation decisions such as follow hallway, turn left or turn right based upon a topological path plan.
5. Finally, input the robot a command such as “SRLT” in your program and place it in the world so that it can execute the path. Use the LEDS to indicate when the robot arrives at the destination. You can send the path to the robot using the GUI and indicate robot status using LEDs, the serial monitor and on the GUI.

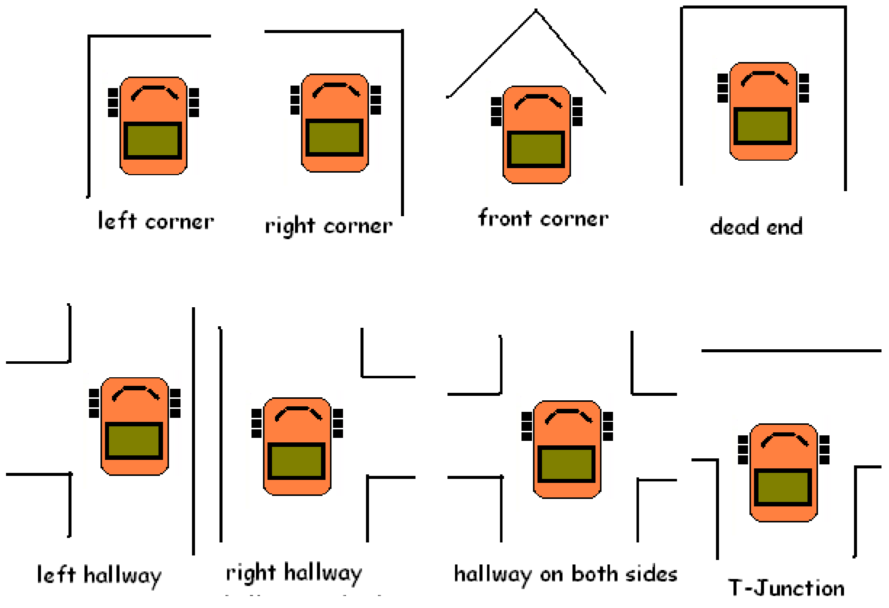


Figure 16: Distinctive Places

6. In the demonstration, you will be required to use this method so that the robot follows several different paths.

Table 3: Perceptual Schema Data Table

Sensor-> Landmark	Front sensor	Left sensor	Right sensor	Back sensor	Front Left	Front Right	Back Left	Back Right
Corner in front								
Corner on left								
Corner on right								
Hallway on both sides								
Hallway on								

left								
Hallway on right								
T-junction								
Dead End								

Part B.2 - Metric Path Planning

1. Download the world map from the Resources project folder in Moodle. The world map is a text file that includes a 4 x 4 array of 0's and 99's that represents free space and obstacles (occupancy grid) or a 4 x 4 array of numbers 0 through 15 that represents world features (topological map). You must devise a method to encode the a priori map into your program. See Figures 5 and 6 for examples of occupancy grids and see Figure 10 for example of topological maps. You should also display the map on the serial monitor to confirm that it matches the real world.

2. At the beginning of the demonstration, you will be given the robot's start position and goal position. Input the start and goal position into your code.

3. The robot should then plan a path from the start position to goal position by using the wavefront algorithm. The serial monitor should display the path planned by the robot. One suggestion for doing this is to show a list of cells that the robot will traverse from the start point to goal point. You can devise a way to indicate this information by using LEDs.

4. You should then place your robot at the start position and it should move to the goal point. You can display the robot's progress by using LEDs to indicate gateways and when the robot has completed the path. You should also use the GUI to send and receive data.

5. Your algorithm should include a combination of odometry and reactive behaviors to avoid hitting the walls while executing the planned path. *(Note that one technique to prevent the robot from hitting walls and obstacles is to select the path that maximizes*

the distance between walls and obstacles or follow the center line (i.e. Voronoi diagram, follow hallway).)

6. In the demonstration, you will be required to show that the robot can follow various distinct paths. You will be graded on how well your algorithm works; the efficiency of the path chosen by the robot, the ability of the robot to reach the goal point while also avoiding obstacles.

Question to answer in the final report:

- What method did you use to send path and goal information to the robot?
- What method did you use to display robot status during path planning and following?
- Create a software design plan for how you will implement topological and metric path planning and path following on your robot.

PART C – LOCALIZATION AND MAPPING

Part C.1 - Localization

1. The user will provide the robot with an a priori map as a matrix to input into your code. The map will be a 4 x 4 array with topological map or occupancy grid encoding.
2. The localization algorithm should process this map to identify key features such as the gateways or distinctive places.
3. The first test of your algorithm will involve using teleoperation to drive your robot through the world until it localizes. Where the robot believes it is in the world should be communicated to the user by using blinks on LEDS or display on the laptop serial monitor using wireless communication.
4. Once the algorithm is working, place the robot in the world. The robot should then use some motion algorithm such as random wander, follow center, or wall following to explore the world and identify gateways. It should keep track of the gateways passed and the order in which they were passed and use it to localize itself. The robot should then send its location to the user by using LEDs or display on the serial monitor.

5. Lastly, after determining its location using registration, the robot should use wavefront propagation to plan a path from its current location to the goal location (or home). The robot's goal will be specified at run time, this information can be transmitted wirelessly to the robot. After localization, you can use prior code to plan a path for the robot.

Part C.2 - Mapping

1. Place the robot in the artificial world and use teleoperation to create a complete occupancy grip or topological map. Use wireless communication, to display the robot's progress and the complete map to the user.
2. Once the initial mapping algorithm is working, place the robot in the artificial world and use motion behaviors to create an occupancy grid or topological map. Use wireless communication, to display the robot's progress and the complete map to the user. You can use LEDs to indicate robot state.
3. Lastly, after creating the map, the robot should localize itself in the map and plan a path to return the robot to its' home or start position. This information will be transmitted wirelessly to the robot. After localization, you can use prior code to plan a path for the robot.

Question to answer in the final report:

- Submit a plan for how you will implement localization and mapping on your robot in the appendix of the report, describe how it worked.
- What method did you use to receive localization data?
- What method did you use to display the world map?

Part D – Simultaneous Localization and Mapping

1. The robot will be placed in an unknown world at an unknown location.
2. The robot must use an autonomous coverage algorithm to create a map of the world and display it on the user interface.

3. The robot will then navigate through the world to find a light beacon in an unknown location. Alternately the robot can use the HuskyLens camera to identify distinct beacons in the world. These should then be displayed on the created map.
4. Once located, the beacon location must be displayed on the user interface.
5. The robot must then explore the world to find the world exit as identified by a location where a wall has mysteriously disappeared.
6. The robot should display status updates as it localizes, creates the map, find the beacon, and plans the escape route and exits the maze.

Question to answer in the final report:

- Submit a plan for how you will implement SLAM on the robot in the appendix of the report and describe how it worked in the method section of the final report.
- How is this different than what you did to implement localization and mapping separately?

****

SUBMISSION REQUIREMENTS

****

Final Project Code:

The final project code should be properly commented and modular with each new behavior representing a new function call. Recall that properly commented code has a header with the solution name, team members' names, description of the functionality and key functions, revision dates. In addition, all of the key variables and functions in the code are commented. The design of the architecture should be evident from the program layout. You should also include the design of the architecture in the appendix of the report and reference it in the text. You must submit your properly commented by midnight on **Sunday of Finals week**.

Final Project Demonstrations:

Due to the complexity of this project and the fact that it requires an integration of several concepts, there will be **three (or four)** software design plans and demonstrations required on a graduated grading scale. You must have all parts of the final project demonstrated by **the last day of class**. The demonstration due dates are given in Table 4.

Final Project Report

Please use the following checklist to insure that your final project report meets the minimum guidelines. You should also view the sample final project reports available on the Moodle course site. You must submit your final project report by midnight on **Sunday of Finals week**.

Project Report Guidelines

1. The document should have default Word settings with respect to font and margins
2. All pages should be numbered
3. All headings must be numbered, left-justified, bolded, and capitalized at the beginning of the section.
4. All figures must have a number and caption underneath (i.e. GUI screenshots)
5. All tables must have a number and title above it (i.e. results error analysis)
6. The report should order should be:

Cover page

Abstract

Table of Contents

I. Objective

II. Theory

III. Methods

IV. Results

V. Conclusions and Recommendations

References

Appendix/Supplementary Materials

7. The ***cover page*** should include the school, course number, course title, team member names, robot name, project title, and date submitted.
8. The ***abstract*** should be a brief statement of the experiment purpose, verification and relevant results. The abstract should be on a separate page after the cover page and before the Table of Contents.
9. The ***table of contents*** should be on a separate page after the abstract and should have page numbers.
10. The ***objective*** should state the purpose of the project and associated tasks in your own words and should be detailed for the reader to understand what the robot must accomplish.
11. The ***theory*** should state relevant theory that will be used to achieve the objective. You must cite relevant theory from the text and other sources and there must be citations and references.
12. The ***methods*** section should summarize the algorithms implemented to achieve the purpose and the procedures used to test the robot algorithms. The method must include a control architecture, flowchart, pseudocode, state diagrams for implementing each part of the objective.
13. The ***results*** section should summarize the results of the tests and verify that the robot was able to achieve the purpose and meet the project objectives. The results should also address the results of any parts that were unsuccessful.
14. The ***conclusions and recommendations*** should address whether the purpose was achieved, possible sources of error, recommendations to improve the robot algorithm and answer any relevant questions related to the project.
15. The ***references*** should include a list of all of the works cited with proper APA or MLA format. Use the Purdue OWL website for more help on proper formatting: https://owl.english.purdue.edu/owl/

16. The ***appendix*** should include all relevant graphics or content that is too dense for the body of the report including flowcharts, control architectures, state diagrams, or pseudocode.

17. Remember this is only a guide for the minimum requirements of your report. You are required to answer any questions or provide any details that you feel aid the reader in understanding the objective, theory, procedure, implementation and results of your project.

18. You should review the sample reports on Moodle for help with the proper format for the document.

Question to answer in the final report (results section, or appendix if appropriate):

1. Were there any issues with the wireless communication? How could you resolve them? If at all.
2. What does the state machine, subsumption architecture, flowchart, or pseudocode look like for the path planning, localization, and mapping? (It should be in the appendix of the report).
3. How would you implement SLAM on the CEENBot given what you have learned about navigation competencies after completing the final project? If you research solutions, make sure you cite and list references in APA or MLA format.
4. What was the strategy for implementing the wavefront algorithm?
5. Were there any points during the navigation when the robot got stuck? If so, how did you extract the robot from that situation?
6. How long did it take for the robot to move from the start position to the goal?
7. What type of algorithm did you use to selection the most optimal or efficient path?
8. How did you represent the robot's start and goal position at run time?
9. Do you have any recommendations for improving that robot's navigation or wavefront algorithm?

10. How did you use the serial monitor and bi-directional wireless communication to represent the map?
11. What type of map did you create and why?
12. What was key in the integration of the localization, mapping, and path planning?

Grading

The final project grade will be based upon the number of tasks in Table 4 that your team successfully completes. ***You must do both parts of the shaded portion to get credit (i.e. you cannot just submit a bunch of software design plans). Note that it is possible to get more than 15 points on the demonstrations.***

There will be rolling demonstrations over the three weeks so as soon as a team has a part ready, they should go ahead and get it graded. All demonstrations, code and final report must be submitted by midnight the Sunday of Finals week.

See examples of some of these demonstrations on the YouTube playlist:

https://www.youtube.com/playlist?list=PL175eO9NwPXKHllu4aVAY3UvTBrS2A3mQ

Table 4: Project Grade Demonstration Point Distribution (15 pts required)

Task	Points
Light Tracking (fear, love, explorer, aggression)	1
Light Tracking State Machine (random wander, avoid obstacle)	2
Hybrid Control with wall following, path planning, path following, docking, light tracking	3
Wireless Communication	3
GUI working with bidirectional communication with robot, robot can be plugged into the laptop but show sending commands and getting sensor updates	3
Topological Path Following	2
metric path planning and following on occupancy grid	4
metric path planning and following on topological map	4
Localization and path planning/following on occupancy grid	4
Localization and path planning/following on topological map	4
Mapping on occupancy grid	4
Mapping on topological map	4
SLAM Escape - Use sensors to create map and localize in a world, identify location of a beacon, recognize where wall is missing to escape and display location of beacon (occupancy grid)	5
SLAM Escape - Demonstration - Use sensors to create map and localize in a world, identify location of a beacon, recognize where wall is missing to escape and display location of beacon (topological map)	5
Software Design Plans	5
Code	10
Report	20